3. PSEUDOCODIFICA DI UN ALGORITMO

In informatica per descrivere un algoritmo vengono utilizzati i seguenti due **linguaggi formali**:

a) un linguaggio, basato sull'utilizzo di simboli grafici, per descrivere i singoli passi dell'algoritmo, detto **linguaggio dei diagrammi a blocchi o flow chart** (oppure **diagramma di flusso**);
b) un linguaggio, molto vicino al linguaggio naturale, detto **linguaggio di progetto o pseudolinguaggio** talvolta abbastanza vicino ad alcuni linguaggi di programmazione.
La descrizione dell'algoritmo effettuata con lo pseudolinguaggio prende il nome di **pseudocodifica**.

La **pseudocodifica** è una fase intermedia che si frappone tra la fase di analisi/progettazione del problema e quella di codifica in un vero e proprio linguaggio di programmazione.
Lo scopo principale della pseudocodifica è di portare l'utente ad esprimere le proprie istruzioni in una forma naturale, utilizzando frasi ed espressioni elementari della lingua italiana che però siano univocamente interpretabili.
Ciò permette di concentrarsi sulla soluzione logica del problema invece che sulla forma e sui vincoli da rispettare nella sua enunciazione.

Valgono le seguenti regole generali:

1) le **parole chiave** o **riservate** saranno scritte in maiuscolo;
2) gli **identificatori** (ossia i nomi) delle **variabili** saranno scritti sempre in *minuscolo* e quelli delle costanti sempre in *maiuscolo*;
3) le parole racchiuse tra **parentesi angolari** < > rappresentano le categorie sintattiche ossia elementi generali del linguaggio che nei vari algoritmi saranno sostituiti con opportune occorrenze;
4) i blocchi racchiusi tra **parentesi quadre** [ ] indicano l'opzionalità ossia tali blocchi possono anche non essere presenti;
5) i blocchi separati dal simbolo | possono essere usati in alternativa (esclusiva).

Lo **pseudocodice** di per sé non può essere caricato direttamente in un calcolatore per essere eseguito ma prima dovrà essere tradotto in un linguaggio di programmazione di alto livello (quali ad esempio il C, il PASCAL, JAVA, .) che andrà poi tradotto in codice binario, l'unico linguaggio che possa essere realmente "capito" dal computer.

I **vantaggi** connessi con l'attività di pseudocodifica sono grandissimi tra i quali segnaliamo:

- la possibilità di **comunicare la logica** dell'algoritmo a persone che non sono dei tecnici;

- la possibilità di **esprimere la logica** senza le limitazioni ed i vincoli tipici dei linguaggi di programmazione;

- la possibilità di **usare lo stesso pseudocodice** per passare poi a diversi linguaggi di programmazione.

Spesso le istruzioni dello pseudocodice vengono indicate con il termine di **pseudoistruzioni**.

SIMBOLOGIA PER LA PSEUDOCODIFICA DI UN ALGORITMO

Struttura della PSEUDOCODIFICA dell'ALGORITMO PRINCIPALE

ALGORITMO <Nome dell'algoritmo>
<AMBIENTE GLOBALE dell'algoritmo: sezione dichiarativa oggetti GLOBALI >
PROCEDURA main( )
<AMBIENTE LOCALE: sezione dichiarativa oggetti LOCALI al sottoprogramma >
INIZIO

<corpo dell'algoritmo>

[**RITORNA**]
FINE

TIPI DI DATO SEMPLICE

INT	per numerico intero
REAL	per numerico reale
CHAR	per singolo carattere alfanumerico
BOOL	per valore logico (VERO o FALSO)

TIPI DI DATO STRUTTURATO

ARRAY[<dimensione>**] DI <Tipo_dato>**
RECORD.....FINE RECORD

OPERATORI ARITMETICI

+	per l'addizione
-	per la sottrazione
*	per la moltiplicazione
/	per la divisione
%	per il resto (modulo n)
DIV	per il quoziente (modulo n)

OPERATORI DI CONFRONTO

=	per "*uguale a*"
<	per "*minore di*
≤	pr "*minore o uguale a*"
>	per "*maggiore di*"
≥	per "*maggiore o uguale a*"
≠	per "*diverso da*"

OPERATORI LOGICI

AND	per la CONGIUNZIONE logica
OR	per la DISGIUNZIONE logica
NOT	per la NEGAZIONE logica

COMMENTO

/* *<qui si scrive la riga*
di commento anche
andando a capo> */

oppure

// *<qui si scrive*
// *la riga di commento*
// *riga x riga>*

A) ISTRUZIONI OPERATIVE

A.1 ISTRUZIONE DI DICHIARAZIONE DELLE VARIABILI

<nome_var1> [, <nome_var2>, <nome_var3>,.....] : <tipo_var>

n.b. Da inserire PREFERIBILMENTE nell'AMBIENTE LOCALE di ogni procedura/funzione

A.1 ISTRUZIONE DI DICHIARAZIONE DELLE COSTANTI

<nome_costante> <valore_costante>

n.b. Da inserire OBBLIGATORIAMENTE nell'AMBIENTE GLOBALE

A.2 ISTRUZIONI DI I/O (input/output)

Leggi (<variabile>) per acquisire un dato dall'esterno

Scrivi (<variabile> | <costante>) per comunicare un dato all'esterno

A.3 ISTRUZIONE DI ASSEGNAZIONE

<variabile > ← **<variabile> | <costante> | <espressione>**

B) ISTRUZIONI DI CONTROLLO

B.1 ISTRUZIONE di SEQUENZA

```
.....
INIZIO
<B1>
<B2>
……
<BN>
FINE
```

<B1>, <B2> …<BN> sono **blocchi semplici** o **blocchi composti** di istruzioni.
Un *blocco semplice* di istruzioni è formato da UNA SOLA istruzine operativa o di controllo.
Un *blocco composto* di istruzioni è invece formato da PIU' blocchi semplici.

B.2 ISTRUZIONI di SELEZIONE (o ISTRUZIONI SELETTIVE)

- Istruzione di SELEZIONE ad una scelta o **UNARIA**

```
.....
SE (<condizione>)
    ALLORA
        <B1>
FINE SE
.....
```

<condizione> è un qualunque enunciato (semplice o composto) dell'Algebra di Boole
<B1> è un **blocco semplice** o **blocco composto** di istruzioni
Se la condizione è VERA si esegue il blocco di istruzioni <B1>
Se la condizione è FALSA si prosegue normalmente**.**

- Istruzione di SELEZIONE a due scelte o **BINARIA**

```
.....
SE (<condizione>)
    ALLORA
        <B1>
    ALTRIMENTI
        <B2>
FINE SE
.....
```

<condizione> è un qualunque enunciato (semplice o composto) dell'Algebra di Boole
<B1> e <B2> sono **blocchi semplici** o **blocchi composti** di istruzioni
Se la condizione è VERA si esegue il blocco di istruzioni <B1>
Se la condizione è FALSA si esegue il blocco di istruzioni <B2>

- Istruzione di SELEZIONE a più scelte o N-**ARIA**

.....
NEL CASO CHE (<variabile> | <espressione>) **SIA**
< valore_1> **:** <B1>
< valore_2> **:** <B2>
…… : ….
< valore_N> **:** <BN>
[ALTRIMENTI : <BN>]
FINE CASO
.....

<*variabile*> | <*espressione*> può essere una variabile oppure una espressione che può assumere un unico valore numerico intero ma non può mai essere espresso come risultato di una condizione logica.
<*valorex*> può essere
- una costante di uguale valore a <variabile> | <espressione>;
- un valore unico;
- una lista di valori;

N.B. E' opportuno ricordare che <valorex> devono essere dello stesso tipo di <variabile> | <espressione>

N.B. Se per più valori distinti devo eseguire lo stesso blocco di istruzioni allora scriverò:

.....
NEL CASO CHE (<variabile> | <espressione>) **SIA**
<valore 1> **:**
<valore 2> :
<valore 3> : <B1>
<valore 4> **:** <B2>
…… : …
< valori n> **:** <BN>
[ALTRIMENTI : <BN+1>]
FINE CASO
.....

B.3 ISTRUZIONI DI ITERAZIONE (o ISTRUZIONI ITERATIVE o CICLICHE)

- Istruzione di ITERAZIONE PRE-CONDIZIONALE

.....
MENTRE (<condizione>) **ESEGUI**
<B1>
FINE MENTRE
.....

<condizione> è un qualunque enunciato (semplice o composto) dell'Algebra di Boole.
<B1> è un **blocco semplice** o un **blocco composto** di istruzioni**.**
Se la condizione è VERA si esegue il blocco di istruzioni <B1>
Se la condizione è FALSA si arresta il processo iterativo.
Se la condizione è inizialmente FALSA il ciclo non viene mai eseguito

- Istruzione di ITERAZIONE POST-CONDIZIONALE

.....
RIPETI
<B1>
FINCHE' (<condizione>)
.....

<condizione> è un qualunque enunciato (semplice o composto) dell'Algebra di Boole.
<B1> è un **blocco semplice** o un **blocco composto** di istruzioni.
Se la condizione è FALSA si esegue il blocco di istruzioni <B1>.
Se la condizione è VERA si arresta il processo iterativo.
Il blocco <B1> di istruzioni viene eseguito ALMENO una volta perché la condizione viene testata dopo la sua esecuzione.

- Istruzione di ITERAZIONE ENUMERATIVA (con numero di iterazioni noto a priori)

.....
PER <indice> ← <inizio> **[INDIETRO] A** <fine> **ESEGUI**
<B1>
<indice> ← <indice> + 1
[<indice> ← <indice> - 1]
FINE PER
.....

La <condizione> è SOTTOINTESA ed è il seguente enunciato dell'Algebra di Boole:

- **<indice> <= <fine>** se il PER è incrementale (o CRESCENTE)
- **<indice> >= <inizio>** se il PER è decrementale (o DECRESCENTE)

<B1> è un **blocco semplice** o un **blocco composto** di istruzioni**.**
Se la condizione SOTTOINTESA è VERA si esegue il blocco di istruzioni <B1>
Se la condizione SOTTINTESA è FALSA si arresta il processo iterativo.
Se la condizione SOTTINTESA è inizialmente FALSA il ciclo non viene mai eseguito

SOTTOPROGRAMMI (in caso di sviluppo modulare dell'algoritmo)

1) PROCEDURE

PROCEDURA <Nome_procedura> **[(REF | VAL** <Nome_param1>**:** <Tipo_param1> **,**
REF | VAL <Nome_param2>**:** <Tipo_param2> ,
……..
REF | VAL <Nome_paramN>**:** <Tipo_paramN> **)]**

< AMBIENTE LOCALE: sezione dichiarativa oggetti locali alla PROCEDURA>

INIZIO

< corpo della procedura>

RITORNA
FINE

2) FUNZIONI

FUNZIONE <Nome_funzione> **[(REF | VAL** <Nome_param1>**:** <Tipo param1> **,**
REF | VAL <Nome_param 2>**:** <Tipo param2> ,
……..
REF | VAL <Nome_paramN>**:** <Tipo paramN> **)]: < Tipo_Risultato>**

< AMBIENTE LOCALE: sezione dichiarativa oggetti locali alla FUNZIONE>

INIZIO

< corpo della funzione>

RITORNA <risultato>

FINE

I FLOW CHART (o Diagrammi di Flusso o Diagrammi a Blocchi)

I **flow chart** sono schemi/disegni che descrivono visivamente come procede l'esecuzione di un algoritmo.
I **flow chart** non sono legati ad uno specifico linguaggio di programmazione: dato un flow chart, il programmatore poi potrà tradurlo usando un qualsiasi linguaggio di programmazione di alto livello.
Il **flow chart** aiuta anche il progettista/programmatore a descrivere correttamente un algoritmo (il procedimento risolutivo di un problema).

Ogni tipo di istruzione che si può inserire in un algoritmo/programma ha un suo simbolo ed ognuna delle tre strutture fondamentali della programmazione (sequenza, selezione ed iterazione) può essere rappresentata.
Esistono anche simboli speciali (inizio programma, fine programma ecc.) che non rappresentano istruzioni vere e proprie ma che sono utili per la costruzione del flow chart.

N.B. Non è previsto alcun simbolo per la dichiarazione di variabili e/o costanti

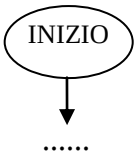
I PRINCIPALI SIMBOLI

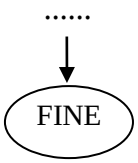
Linea di flusso ***FRECCIA orizzontale o verticale***

Gruppo di istruzioni che precedono o seguono

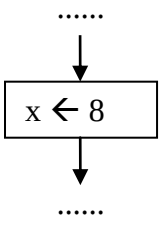
.....

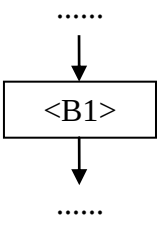
Inizio/Fine programma ***ELLISSE***





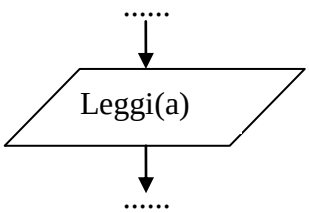
Assegnazione o altre istruzioni generiche (blocchi di istruzioni) ***RETTANGOLO***
(esempio: dai ad x il valore 8)



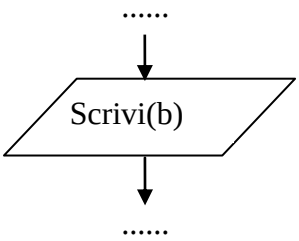


(esempio: rappresenta un generico blocco semplice o composto di istruzioni)

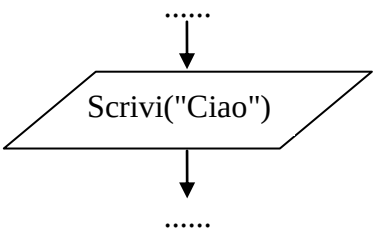
Lettura da tastiera del valore scritto da memorizzare nella variabile indicata tra parentesi tonde
PARALLELOGRAMMA
(esempio: leggi il valore della variabile di INPUT a*)*



Scrittura a video del valore memorizzato in una variabile o in una costante
(esempio: scrivi il valore della variabile di OUTPUT b)



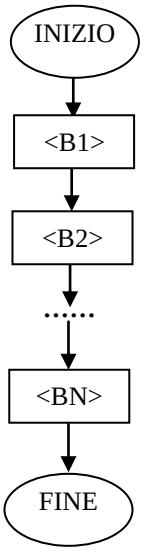
(esempio: scrivi a video la stringa COSTANTE "Ciao"))



I simboli visti fino ad ora servono per istruzioni singole. Vediamo ora come si rappresentano le strutture fondamentali della programmazione (sequenza, selezione, iterazione) ossia le

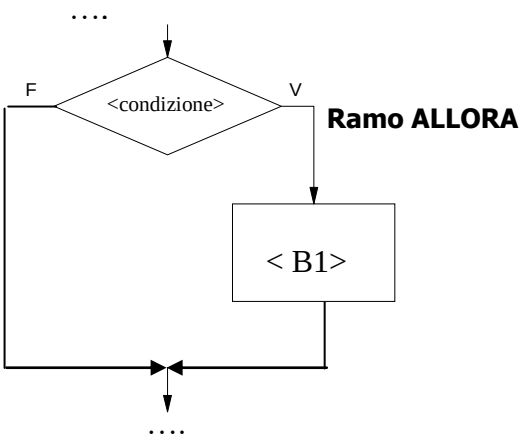
ISTRUZIONI DI CONTROLLO

Istruzione di SEQUENZA

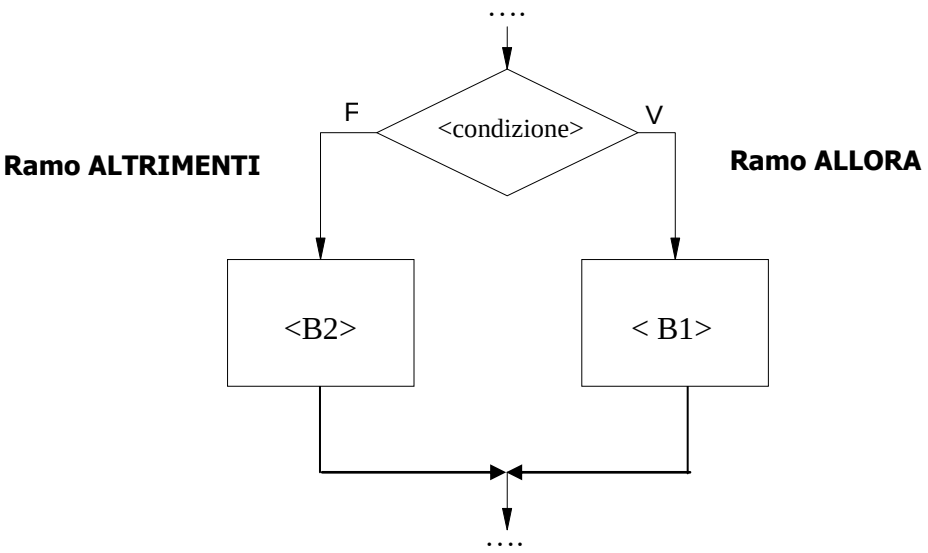


Istruzione di SELEZIONE

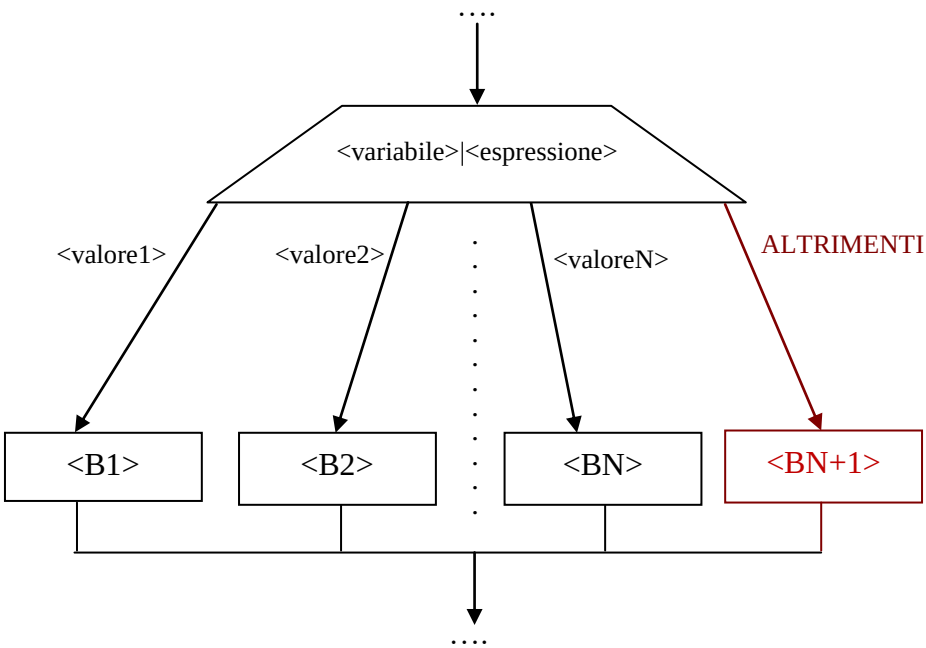
Istruzione di SELEZIONE a UNA VIA o UNARIA



Istruzioni di Selezione a DUE VIE o BINARIA

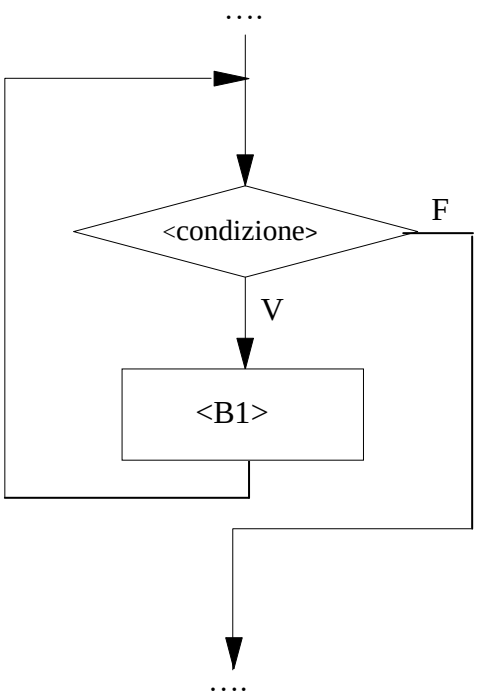


Istruzione di SELEZIONE a N VIE o ENNARIA

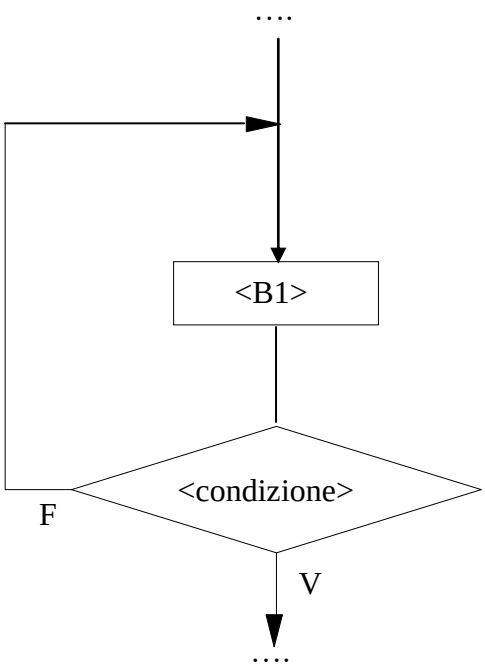


Istruzione di ITERAZIONE

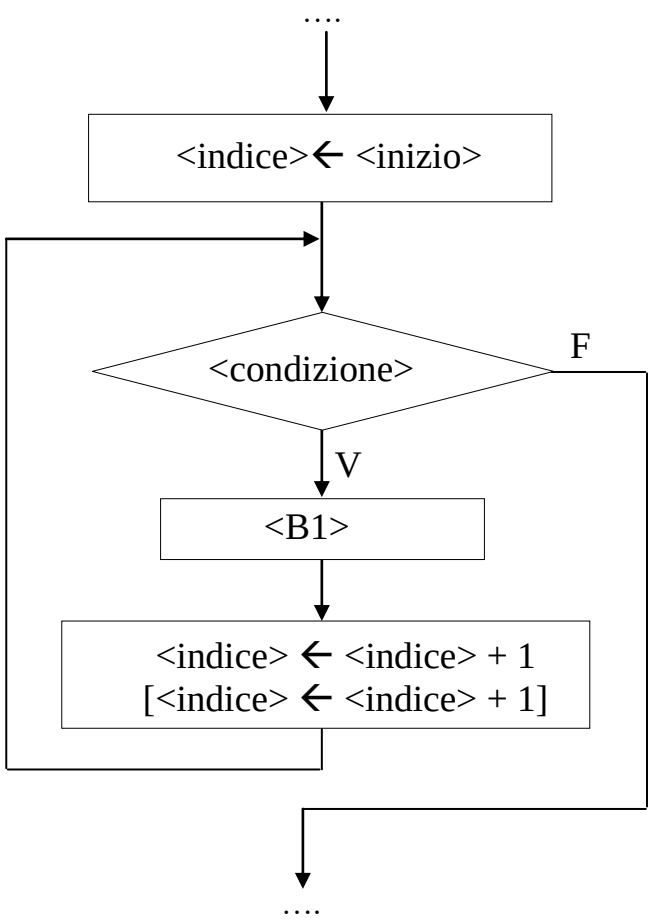
Istruzione di ITERAZIONE PRE-CONDIZIONALE che cicla per valore VERO

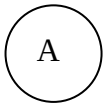
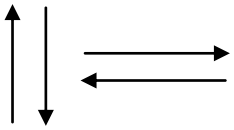
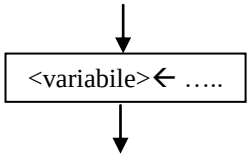
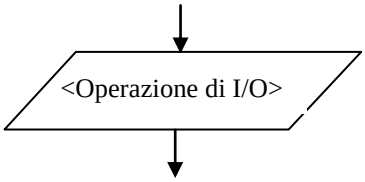
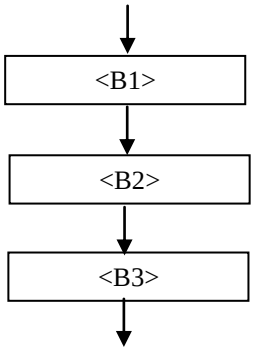
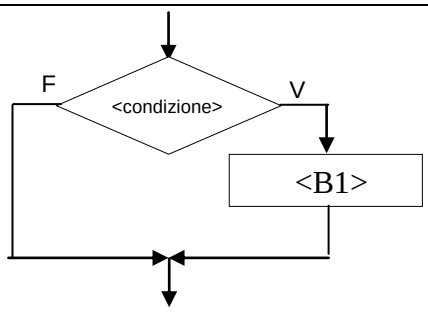


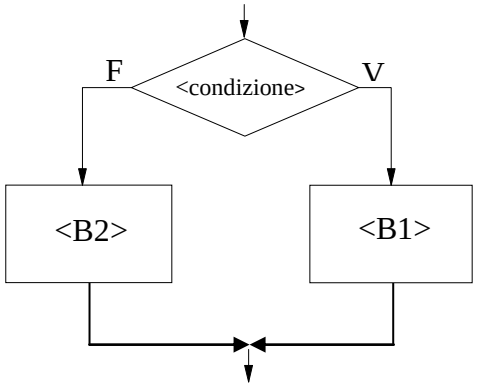
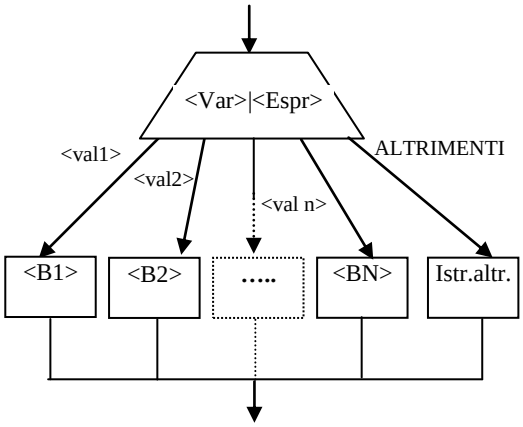
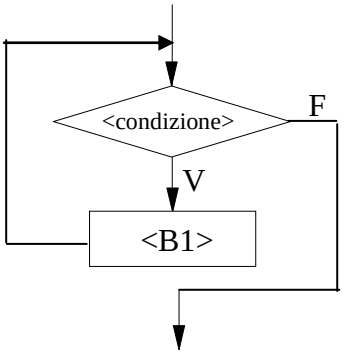
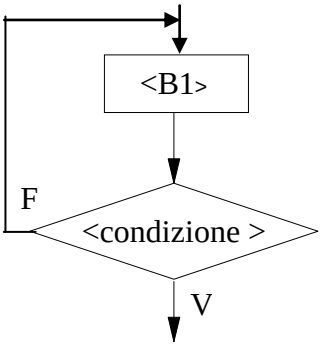
Istruzione di ITERAZIONE POST-CONDIZIONALE che cicla per valore FALSO



Istruzione ITERAZIONE ENUMERATIVA (con numero di iterazioni noto a priori)



Azioni dell'ALGORITMO da rappresentare	PSEUDOISTRUZIONI della PSEUDOCODIFICA	Simboli del FLOW CHART O del DIAGRAMMA A BLOCCHI
Inizio	**INIZIO**	
Fine	**FINE**	
Connettore	Indica l'inizio o la fine di una parte di diagramma che è stato diviso per comodità di lettura. Contiene al'interno un simbolo o un numero che corrisponde ad un altro connettore	
Linee di flusso	Indica il flusso dei dati all'interno del processo risolutivo dell'algoritmo	
Istruzione di Assegnazione (ISTRUZIONE OPERATIVA)	<variabile> ← <variabile> | <costante> | <espressione>	
Istruzione di I/O da tastiera o su video (ISTRUZIONE OPERATIVA)	**Leggi** (<variabile>) oppure **Scrivi** (<variabile> | <costante >)	
Istruzione di "SEQUENZA" (ISTRUZIONE DI CONTROLLO)	…… <B1> <B2> <B3> ……	
Istruzione di "SELEZIONE UNARIA o A UNA VIA" (ISTRUZIONE DI CONTROLLO)	**SE** <condizione> **ALLORA** <B1> **FINE SE**	

Azioni dell'ALGORITMO da rappresentare	**PSEUDOISTRUZIONI della PSEUDOCODIFICA**	**Simboli del FLOW CHART O del DIAGRAMMA A BLOCCHI**
Istruzione di "SELEZIONE BINARIA o A DUE VIE" (ISTRUZIONE DI CONTROLLO)	**SE** <condizione> **ALLORA** <B1> **ALTRIMENTI** <B2> **FINE SE**	
Istruzione di "SELEZIONE ENNARIA o a N VIE" (ISTRUZIONE DI CONTROLLO)	**NEL CASO CHE (**<var>>|<espr>) **SIA** < valore 1> **:** <blocco istruzioni 1> < valore 2> **:** <blocco istruzioni 2> …… : …… < valore N> **:** <blocco istruzioni N> **[ALTRIMENTI** : <blocco istruzioni altri casi>**]** **FINE CASO**	
Istruzione di ITERAZIONE PRE-CONDIZIONALE (ISTRUZIONE DI CONTROLLO)	**MENTRE** (<condizione>) **ESEGUI** < B1> **FINE MENTRE**	
Istruzione di ITERAZIONE POST-CONDIZIONALE (ISTRUZIONE DI CONTROLLO)	**RIPETI** < B1> **FINCHE' (** <condizione>)	

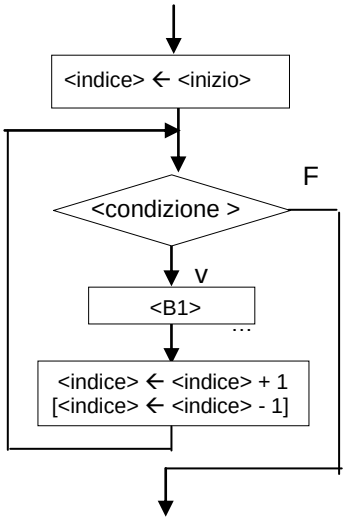
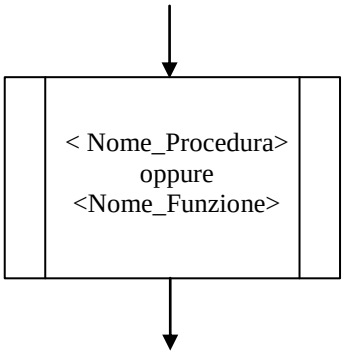
Azioni dell'ALGORITMO da rappresentare	**PSEUDOISTRUZIONI della PSEUDOCODIFICA**	**Simboli del FLOW CHART O del DIAGRAMMA A BLOCCHI**
Istruzione di ITERAZIONE ENUMERATIVA (ISTRUZIONE DI CONTROLLO)	**PER** <indice> ← <inizio> **[INDIETRO] A** <fine> **ESEGUI** < B1> <indice> ← <indice> + 1 **[**<indice> ← <indice> - 1**]** **FINE PER** Dove la <condizione> logica sottintesa è: • **<indice> <= <fine>** se il PER è incrementale • **<indice> >= <inizio>** se il PER è decrementale	
Programma principale	**ALGORITMO** <Nome dell'algoritmo> **PROCEDURA** main( ) <seziond dichiarativa locale alla procedura main ( )> **INIZIO** <corpo della procedura main ( )> **FINE**	E' l'insieme di tutti i simboli grafici visti finora
Istruzione di chiamata a sottoprogramma (gruppo di istruzioni che verranno elaborate insieme)	**PROCEDURA** <Nome_procedura.> **([REF | VAL** <Nome_param1>**:** <Tipo_param1> **,** **REF | VAL** <Nome_param2>**:** <Tipo_param2> , …….., **REF | VAL** <Nome_paramN>**:** <Tipo_paramN> **])** < sezione dichiarativa locale alla procedura> **INIZIO** < corpo della procedura> **RITORNA** **FINE** **FUNZIONE** <Nome_funzione> **([REF | VAL** <Nome_param1>**:** <Tipo_param1> **,** **REF | VAL** <Nome_param2>**:** <Tipo_param2> , …….., **REF | VAL** <Nome_paramN>**:** <Tipo_paramN> **])** **: < Tipo Risultato>** < sezione dichiarativa locale alla funzione> **INIZIO** < corpo della funzione> **RITORNA** <risultato> **FINE**	

TABELLE RIASSUNTIVE PER L'ANALISI DEI DATI

DATI DI INPUT DEL PROBLEMA PRINCIPALE (PROCEDURA MAIN)				
Nome variabile (1)	**Tipo dati (2)**	**Tipo Allocazione (3)**	**Valori ammessi (4)**	**Descrizione (5)**
DATI DI OUTPUT DEL PROBLEMA PRINCIPALE (PROCEDURA MAIN)				
Nome variabile	**Tipo dati**	**Tipo Allocazione**	**Valori ammessi**	**Descrizione**
DATI DI ELABORAZIONE (LAVORO) DEL PROBLEMA PRINCIPALE (PROCEDURA MAIN)				
Nome variabile oppure **nome costante (6)**	**Tipo dati**	**Tipo Allocazione**	**Valori ammessi**	**Descrizione**

(1) Il nome di una variabile deve essere scritto **tutto con caratteri minuscoli** senza utilizzare spazi e caratteri speciali (quali &, *,#, @). In caso di nomi di variabili costituiti da più di una parola è possibile utilizzare il carattere '-' (trattino) oppure '_' (underscore).
Esempio: Sono nomi di variabili valide i seguenti: *a, pippo, somma-2, somma_2*
Sono nomi di variabili non valide i seguenti: A*, Pippo, somma 2, somma$2*

(2) I tipi utilizzabili sono i tipi di dato semplice (un solo valore possibile per una variabile in un certo istante durante 'esecuzione dell'algoritmo) o tipo di dato strutturato o struttura dati (più valori possibili per una variabile in un certo istante durante l'esecuzione dell'algoritmo)
I tipi di dato semplice che utilizzeremo sono:
INT: per i numeri interi relativi (eventualmente con il segno) ma senza parte decimale; i valori possibili per questo tipo di variabile sono i normali interi numeri scritti nella modalità consueta;
Esempio*: 12 -45 +32*

REAL: per i numeri reali (eventualmente con il segno) con parte decimale; ; i valori possibile per questo tipo di variabile sono i normali numeri reali scritti nella modalità consueta utilizzando la virgola decimale;
Esempio*: 12,34 -45,00 +32,1 ma anche 12 -45 +32*

CHAR: per il singolo carattere alfanumerico secondo la codifica ASCII estesa: i valori possibili per questo tipo di variabile sono i caratteri racchiusi da due apici singoli;
Esempio*: 'a' 'M' '$' '@' ';' 'Þ'*

BOOL: per il singolo valore di verità dell'algebra booleana: i valori possibili per questo tipo di variabile sono i seguenti: VERO oppure FALSO;

(3) Da valorizzare con:
STATICA per indicare l'allocazione della variabile di tipo statico (ossia costante per tutta la durata dell'algoritmo)
DINAMICA per indicare l'allocazione della variabile di tipo dinamico (ossia modificabile nel corso dell'esecuzione dell'algoritmo)
N.B. per questa prima fase useremo variabili esclusivamente STATICHE per le quali, quindi, occorre valorizzare con **STATICA** la colonna "Tipo Allocazione" ad eccezione delle costanti per le quali scriveremo **N.A.** ossia non applicabile nella medesima colonna.

(4) In questa colonna occorre indicare, utilizzando se possibile il linguaggio della matematica, le condizioni che i valori di quella variabile devono rispettare nel corso dell'algoritmo.
Esempio*: 'Se a è la variabile STAT di tipo INT che rappresenta il primo coefficiente di un'equazione di secondo grado è ovvio che dovrà essere scritta, nella colonna "Valori ammessi", la condizione* ***a≠0***

(5) Da valorizzare indicando brevemente ma in modo significativo la spiegazione del significato della variabile.

(6) Il nome di una costante deve essere scritto **tutta con caratteri MAIUSCOLI** senza utilizzare spazi e caratteri speciali (quali &, *,#, @). In caso di nomi di costanti costituiti da più di una parola è possibile utilizzare il carattere '-' (trattino) oppure '_' (underscore).
Esempio: Sono nomi di costanti valide i seguenti: *PIGRECO PI-GRECO PI_GRECO*
Sono nomi di costanti non valide i seguenti: *Pigreco PI+GRECO PI&GRECO*